Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __

Test Date: 3/22/2024

Test Case ID#: Quota_1

Name(s) of Testers: Crystal Wen

Test Description:
"calculateQuota" method:
Test whether the value of the quota will be correctly calculated.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCPL.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:

The total number of seats and votes cannot be zero.
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test calculateQuota for 2 integers that are not divisible.	TestCPL.java testCPLVote.csv Number of votes: 10000 Number of seats: 3	Quota: 3333	Quota: 3333	The result matches the expected result.
2	Test calculateQuota for 2 integers that are divisible.	TestCPL.java testCPLVote.csv Number of votes: 10 Number of seats: 5	Quota: 2	Quota: 2	The result matches the expected result.

Post condition(s) for Test:

The calculated quota is an integer with the value of the floor of the quotient between the total number of votes and the total number of seats.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __

Test Date: 3/22/2024

Test Case ID#: Vote_Count_CPL_2

Name(s) of Testers: Crystal Wen

Test Description:
"voteCounting" method:
Tests whether the vote counting method for a CPL-type election will correctly count votes for each party.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCPL.java

Automated: yes x no

Results: Pass x Fail

Preconditions for Test:

There must be at least 1 ballot in the given file.
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test voteCounting with an ArrayList of parties and compare the expected number of votes to the actual votes for the Democratic party	TestCPL.java testCPLVote.csv Party: Democratic	Democratic: 3	Democratic: 3	The result matches the expected result.
2	Compare the expected number of votes to the actual number of votes for the Republican party.	TestCPL.java testCPLVote.csv Party: Republican	Republican: 2	Republican: 2	The result matches the expected result.
3	Compare the expected number of votes to the actual number of votes for the New Wave party.	TestCPL.java testCPLVote.csv Party: New Wave	New Wave: 0	New Wave: 0	The result matches the expected result.

4	Compare the expected number of votes to the actual number of votes for the Reform party.	TestCPL.java testCPLVote.csv Party: Reform	Reform: 2	Reform: 2	The result matches the expected result.
5	Compare the expected number of votes to the actual number of votes for the Green party.	TestCPL.java testCPLVote.csv Party: Green	Green: 1	Green: 1	The result matches the expected result.
6	Compare the expected number of votes to the actual number of votes for the Independent party.	TestCPL.java testCPLVote.csv Party: Independent	Independent: 1	Independent: 1	The result matches the expected result.

Post condition(s) for Test:

There is the correct number of votes for each party.

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __ **Test Date: 3/22/2024**

Test Case ID#: Coin_Toss_CPL_3 **Name(s) of Testers: Crystal Wen**

Test Description:
"coinToss" method:
Test the fairness of the coinToss method that uses a random integer generator.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCoinToss.java

Automated: yes___ no _x__

Results: Pass x Fail

Preconditions for Test:

A tie between at least two parties must appear when finding a winner.
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Test coinToss for 2 parties. Runs the method 1000 times	TestCoinToss.java testCPLVote.csv Parties: Democratic and Republican	Times of Democratic: around 500 Times of Republican: around 500	Times of Democratic: 509 Times of Republican: 491	The result is not an exact match, but it is within an acceptable range.
2	Test coinToss for more than 2 parties. Runs the method 1000 times	TestCoinToss.java testCPLVote.csv Candidates: Democratic, Republican, and Reform	Times of Democratic: around 333 Times of Republican: around 333 Times of Reform: around 333	Times of Democratic: around 320 Times of Republican: around 359 Times of Reform: around 321	The result is not an exact match, but it is within an acceptable range.

Post condition(s) for Test:

The coin toss method is fair, which means that each party has an almost equal probability of being chosen.

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __ **Test Date:** 3/23/2024

Test Case ID#: Vote_Count_OPL_4 **Name(s) of Testers:** Crystal Wen

Test Description:
"voteCounting" method:
Tests whether the vote counting method for an OPL-type election will correctly count votes for each party and candidate.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestOPL.java

Automated: yes_ x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:

There must be at least 1 ballot in the given file.
"testOPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Test voteCounting with an ArrayList of parties and ArrayList of candidates. It should then compare the expected number of votes to the actual votes for the Democratic party	TestOPL.java testOPLVote.csv Party: Democratic	Democratic: 3	Democratic: 3	The result matches the expected result.
2	Compare the expected number of votes to the actual number of votes for the Republican party.	TestOPL.java testOPLVote.csv Party: Republican	Republican: 4	Republican: 4	The result matches the expected result.
3	Compare the expected number of votes to the actual number of votes for the Independent1 party.	TestOPL.java testOPLVote.csv Party: Indpendent1	Independent1: 2	Independent1: 2	The result matches the expected result.
4	Compare the expected number of votes to the actual number of votes for the candidate Pike	TestOPL.java testOPLVote.csv Candidate: Pike	Pike: 2	Pike: 2	The result matches the expected result.
5	Compare the expected number of votes to the actual number of votes for the candidate Lucy	TestOPL.java testOPLVote.csv Candidate: Lucy	Lucy: 1	Lucy: 1	The result matches the expected result.
6	Compare the expected number of votes to the actual number of votes for the candidate Beiye.	TestOPL.java testOPLVote.csv Candidate: Beiye	Beiye: 0	Beiye: 0	The result matches the expected result.
7	Compare the expected number of votes to the actual number of votes for the candidate Etta.	TestOPL.java testOPLVote.csv Candidate: Etta	Etta: 2	Etta: 2	The result matches the expected result.
8	Compare the expected number of votes to the actual number of votes for the candidate Alawa.	TestOPL.java testOPLVote.csv Candidate: Alawa	Alawa: 2	Alawa: 2	The result matches the expected result.
9	Compare the expected number of votes to the actual number of votes for the candidate Sasha.	TestOPL.java testOPLVote.csv Candidate: Sasha	Sasha: 2	Sasha: 2	The result matches the expected result.

Post condition(s) for Test:

There is the correct number of votes for each party and candidate.

Test Stage: Unit _x_ System __

Test Date: 3/23/2024

Test Case ID#: Coin_Toss_OPL_5

Name(s) of Testers: Crystal Wen

Test Description:
"coinToss" method:
Test the fairness of the coinToss method that uses a random integer generator.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCoinToss.java

Automated: yes no x

Results: Pass x Fail

Preconditions for Test:

A tie between at least two candidates must appear when finding a winner.
"testOPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test coinTossOPL for 2 candidates. Runs the method 1000 times	TestCoinToss.java Candidates: Sara and Bob Mc'Bobson	Times of Sara: around 500 Times of Bob Mc'Bobson: around 500	Times of Democratic: 508 Times of Republican: 492	The result is not an exact match, but it is within an acceptable range.
2	Test coinTossOPL for more than 2 candidates. Runs the method 1000 times	TestCoinToss.java Candidates: Sara, Bob Mc'Bobson, Steve Mc'Steveson, and Renee	Times of Sara: around 250 Times of Bob Mc'Bobson: around 250 Times of Steve Mc'Steveson: around 250 Times of Renee: around 250	Times of Sara: 267 Times of Bob Mc'Bobson: 254 Times of Steve Mc'Steveson: 243 Times of Renee: around 236	The result is not an exact match, but it is within an acceptable range.

Post condition(s) for Test:

The coin toss method is fair, which means that each candidate has an almost equal probability of being chosen.

Test Stage: Unit _X_ System __

Test Date: 3/24/2024

Test Case ID#: Display_Results_6

Name(s) of Testers: Lysong Seang

Test Description:
"displayResults"method:
test whether the results are correctly display

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manually Testing

Automated: yes ___ no X

Results: Pass ___ Fail X

Preconditions for Test: When required information has been provided
Voting algorithm is completed.
The election result is clarified.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check each display line and compare to the expected output	Election Type is CPL	CPL	CPL	The result matches the expected result.
2	Check each display line and compare to the expected output	Election Type is OPL	OPL	OPL	The result matches the expected result.
3	Check each display line and compare to the expected output	Number of Party :6 Number of Seat is 3 Number of Ballots : 9 Number of Candidate : 11	Number of Party : 5 Number of Seat : 3 Number of Ballots : 9 Number of Candidate : 12	Number of Party : 5 Number of Seat : 3 Number of Ballots : 9 Number of Candidate : 12	The result matches the expected result.
4	Check if the quota display correctly	Number of votes: 9 Number of seats: 3	Number of Quota: 3	Number of Quota: 3	The result matches the expected result.
5	Display in CPL style	Total Seats :3 Democratic: Joe, Sally, Ahmed Republican, Allen, Nikki, Taihui New Wave, Sarah Reform, Xinyue, Nikita Green, Bethany	Democratic: Number of Seats: 1 *** Winner(s): Joe, *** Number of Votes: 3 % of votes: 33.33333333333333 Candidate(s): Joe, Sally, Ahmed,	Democratic: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 3 % of votes: 33.33333333333333 Candidate(s): Joe, Sally, Ahmed,	FAIL: It always prints out the same winners instead of the winner corresponding to the party.

		Independent, Mike Number of Democrat seat is 1 Number of Republican seat is 1 Number of Reform seat is 1 Number of New Wave seat is 0 Number of of Independent seat is 0 Number of Democrat votes : 3 Number of Republican votes: 2 Number of Reform votes: 2 Number of New Wave votes: 0 Number of Green vote : 1 Number of Independent : 1	__ Republican: Number of Seats: 1 *** Winner(s): Allen *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Allen, Nikki, Taihui, _____ __ New Wave: Number of Seats: 0 *** Winner(s): N/A*** Number of Votes: 0 % of votes: 0.0 Candidate(s): Sarah, _____ __ Reform: Number of Seats: 1 *** Winner(s): Xinyue *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Xinyue, Nikita, _____ ___ Green: Number of Seats: 0 *** Winner(s): N/A *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Bethany, _____ ___ Independent: Number of Seats: 0 *** Winner(s): N/A *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Mike, _____ ___	Republican: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue*** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Allen, Nikki, Taihui, _____ New Wave: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue*** Number of Votes: 0 % of votes: 0.0 Candidate(s): Sarah, _____ Reform: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Xinyue, Nikita, _____ Green: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue*** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Bethany, _____ Independent: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue*** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Mike, _____	
6	Display in OPL style	Total Seats : 2 Democrat, Pike Democrat, Lucy Democrat, Beiye Republican, Etta Republican, Alawa Independent, Sasha Number of Democrat seat is 1 Number of Republican seat is 1 Number of independent seat is 0	***** Winner ***** 1. Pike (22.22222222222222 / 2) 2. Etta (22.22222222222222 / 2) ***** Candidate ***** Democrat Won: 1 seat(s) Candidate: Pike, Lucy, Beiye Republican Won: 1 seat(s) Candidate: Etta, Alawa Independent1 Won: 0 seat(s) Candidate: Sasha	***** Winner ***** 1. Pike (22.22222222222222 / 2) 2. Etta (22.22222222222222 / 2) ***** Candidate ***** Democrat Won: 1 seat(s) Candidate: Pike, Lucy, Beiye Republican Won: 1 seat(s) Candidate: Etta, Alawa Independent1 Won: 0 seat(s) Candidate: Sasha	The number inside the parenthesis corresponding to percent of number of each candidate gets divided by the total votes and the number of vote each candidate gets respectively (22.22222222222222 / 2)

Post condition(s) for Test: Displays the results after the voting algorithm

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Audit_7 **Name(s) of Testers: Lysong Seang and Shunichi Sawamura**

Test Description:
"audit" method to see if the file produces and the content written to the file is correct.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

Automated: yes___ no _X_ **Manually Tested**

Results: Pass __X___ Fail_____

Preconditions for Test: Required information has been provided.
Voting algorithm is completed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Produce an audit file	auditFile{Date_Time}.txt	auditFile{Date_Time}.txt file produce	auditFile{Date_Time}.txt file produce	The result matches the expected result. The file is always a txt file. The file name is "auditFile" +

					date and time when the program generates an audit file.
2	Test the written content in audit file	Total Seats :3 Democratic: Joe, Sally, Ahmed Republican, Allen, Nikki, Taihui New Wave, Sarah Reform, Xinyue, Nikita Green, Bethany Independent, Mike Number of Democrat seat is 1 Number of Republican seat is 1 Number of Reform seat is 1 Number of New Wave seat is 0 Number of of Independent seat is 0 Number of Democrat votes : 3 Number of Republican votes: 2 Number of Reform votes: 2 Number of New Wave votes: 0 Number of Green vote : 1 Number of Independent : 1	Election type: CPL Number of Parties: 6 Number of Ballots: 9 Number of Seats: 3 Quota Value: 3 Democratic: Joe, Sally, Ahmed, Republican: Allen, Nikki, Taihui, New Wave: Sarah, Reform: Xinyue, Nikita, Green: Bethany, Independent: Mike, ----- Democratic ----- Total Seats: 1 Votes:3 / Quota:3 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 ----- Republican ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- New Wave ----- Total Seats: 0 Votes:0 / Quota:3 = First Allocation Seats:0 Remaining Votes:0 --> Second Allocation Seats:0 ----- Reform ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- Green ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 ----- Independent ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 *** Winner(s) *** Joe (Democratic)	Election type: CPL Number of Parties: 6 Number of Ballots: 9 Number of Seats: 3 Quota Value: 3 Democratic: Joe, Sally, Ahmed, Republican: Allen, Nikki, Taihui, New Wave: Sarah, Reform: Xinyue, Nikita, Green: Bethany, Independent: Mike, ----- Democratic ----- Total Seats: 1 Votes:3 / Quota:3 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 ----- Republican ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- New Wave ----- Total Seats: 0 Votes:0 / Quota:3 = First Allocation Seats:0 Remaining Votes:0 --> Second Allocation Seats:0 ----- Reform ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- Green ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 ----- Independent ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 *** Winner(s) *** Joe (Democratic) Allen (Republican)	The Election Type could change to “OPL” depending on the file.

			Allen (Republican) Xinyue (Reform)	Xinyue (Reform)	

Post condition(s) for Test:

An audit file containing statistics and information about the election is created

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __

Test Date: 3/24/2024

Test Case ID#: Candidate_Initialization_8

Name(s) of Testers: Fumisato Teranishi

Test Description:
"Candidate" class and the "getName", "getParty", "getNumVotes" methods:
Tests if a Candidate object is properly initialized.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCandidate.java

Automated: yes x no

Results: Pass __x___ Fail_____

Preconditions for Test:
The Candidate must have a name, party, and a number of votes greater than or equal to 0

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Candidate object and gets the name of the Candidate	TestCandidate.java	Name: John Doe	Name: John Doe	The expected results match the actual results

2	Compare the Candidate's party to what is returned by getParty()	TestCandidate.java	Party: Independent	Party: Independent	The expected results match the actual results
3	Compare the Candidate's party to what is returned by getNumVotes(()	TestCandidate.java	Number of Votes: 1000	Number of Voes: 1000	The expected results match the actual results

Post condition(s) for Test:

The Candidate object has its attributes (name, party, and numVotes) set to the given parameters.

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __

Test Date: 3/24/2024

Test Case ID#: Set_Candidate_Votes_9

Name(s) of Testers: Fumisato Teranishi

Test Description:
"setNumVotes" method:
Test if the method will change the Candidate's number of votes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCandidate.java

Automated: yes_x__ no ___

Results: Pass x Fail

Preconditions for Test:

A Candidate has been initialized and there are votes counted for the Candidate

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Candidate and changes the number of votes after initialization.	TestCandidate.java	Number of votes: 1500	Number of votes 1500	The expected result matches the actual result.

Post condition(s) for Test:

The number of votes of the Candidate was changed to the given number.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __

Test Date: 3/24/2024

Test Case ID#: Party_Initialize_10

Name(s) of Testers: Fumisato Teranishi

Test Description:
"Party" constructor and "getName", "getNumVotes", "getCandidates", and getNumAllocatedSeats":
Test to see if the Party class is properly initialized.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestParty.java

Automated: yes x no

Results: Pass __x___ Fail_____

Preconditions for Test:
The file requires at least one Party and at least one Candidate.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Party object and calls getName() to compare the name attribute.	TestParty.java	Name: Democratic	Name: Democratic	The result matches the expected result.
2	Calls getNumVotes to compare the numVotes attribute.	TestParty.java	Number of Votes: 0	Number of Votes: 0	The result matches the expected result.
3	Calls getCandidates() to compare the candidate attribute	TestParty.java	Candidate 1: Joe, Democratic, 0 Candidate 2: Sally, Democratic, 0 Candidate 3: Ahmed, Democratic, 0	Candidate 1: Joe, Democratic, 0 Candidate 2: Sally, Democratic, 0 Candidate 3: Ahmed, Democratic, 0	The result matches the expected result.
4	Calls getNumAllocatedSeats to compare the numAllocatedSeats attribute	TestParty.java	Number of Allocated Seats: 0	Number of Allocated Seats: 0	The result matches the expected result.

Post condition(s) for Test:

The Party object is properly and correctly initialized.

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Set_Party_Votes_11 **Name(s) of Testers: Fumisato Teranishi**

Test Description:
"setNumVotes" method:
Test if the method will change the number of votes of the party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestParty.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:

The Party object is properly initialized and a Party gained another vote.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Party object and changes the number of votes by calling setNumVotes	TestParty.java	Number of Votes: 100	Number of Votes: 100	The result matches the expected result.
2					

Post condition(s) for Test:

The number of votes of the Party was changed to the given number.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _X_ System __ **Test Date: 3/24/2024**

Test Case ID#: Set_Candidates_12 **Name(s) of Testers: Fumisato Teranishi**

Test Description:
"setCandidates" method:
Test if the method will change the list of Candidates of the party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestParty.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:
A Party object was properly initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Party object and changes the list of Candidates by calling setCandidates	TestParty.java	Candidate 1: Joe, Democratic, 0 Candidate 2: Sally, Democratic, 0 Candidate 3: Ahmed, Democratic, 0	Candidate 1: Joe, Democratic, 0 Candidate 2: Sally, Democratic, 0 Candidate 3: Ahmed, Democratic, 0	The result matches the expected result.

Post condition(s) for Test:
The candidate list of the Party was changed to the given candidate list.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Set_Allocated_Seats_13 **Name(s) of Testers: Fumisato Teranishi**

Test Description:
"setNumAllocatedSeats":
Tests if the method will change the number of allocated seats of the party

Automated: yes x no __ **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
./Project1/src/TestParty.java

Results: Pass x Fail

Preconditions for Test:
A Party object was properly initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initializes a Party object and changes the number of allocated seats by calling setNumAllocatedSeats().	TestParty.java	Number of Allocated Seats: 2	Number of Allocated Seats: 2	The result matches the expected result.

Post condition(s) for Test:
The number of allocated seats of the party was changed to the given number of seats.

Project Name: Project 1: Voting System Team#16

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Allocate_Seats_14 **Name(s) of Testers: Shunichi Sawamura**

Test Description:
"allocateSeats" method:
Tests whether the allocating seats method in Election class will correctly distribute seats to parties based on the largest remainder approach.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCPL.java

Automated: yes_ x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:

There must be at least 1 ballot in the given file.
Since allocateSeats() method is always implemented after completing voteCounting() method, the test runs with the condition that voteCounting() is already completed.
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test allocateSeats with an ArrayList of parties and check the number of allocated seats for Democratic is correct.	TestCPL.java testCPLVote.csv	The number of allocated seats for Democratic: 1	The number of allocated seats forDemocratic: 1	The result matches the expected result.
2	Check if the number of allocated seats for Republican is correct.	TestCPL.java testCPLVote.csv	The number of allocated seats for Republican: 1	The number of allocated seats for Republican: 1	The result matches the expected result.
3	Check if the number of allocated seats for Independent1 is correct.	TestCPL.java testCPLVote.csv	The number of allocated seats for Independent1: 0	The number of allocated seats for Independent1: 0	The result matches the expected result.

Post condition(s) for Test:

The correct number of seats is allocated to each party through allocateSeats().

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Allocate_Seats_Coin_Toss_15 **Name(s) of Testers: Shunichi Sawamura**

Test Description:
"allocateSeats" method:
Tests whether the allocating seats method in Election class will correctly distribute seats to parties and run the coin toss if there is a tie between parties.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCPL.java

Automated: yes_ x__ no ___

Results: Pass x Fail

Preconditions for Test:

There must be at least 1 ballot in the given file.
Since allocateSeats() method is always implemented after completing voteCounting() method, the test runs with the condition that voteCounting() is already completed.
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compare the total number of the allocated seats with total seats given in the election	TestCPL.java testCPLVote.csv	The total number of allocated seats: 3	The total number of allocated seats: 3	The total number of allocated seats is calculated from the sum of allocated seats in each party. The result matches the expected result.
2	Check if the number of allocated seats for Democratic is correct.	TestCPL.java testCPLVote.csv	The number of allocated seats for Democratic: 1 or 2	The number of allocated seats for Democratic: 1 or 2	The result matches the expected result. Since a seat is allocated randomly by coin toss, the result can be expected with two kinds of value.
3	Check if the number of allocated seats for Republican is correct.	TestCPL.java testCPLVote.csv	The number of allocated seats for Republican: 1 or 2	The number of allocated seats for Republican: 1 or 2	The result matches the expected result. Since a seat is allocated

					randomly by coin toss, the result can be expected with two kinds of value.

Post condition(s) for Test:

The correct number of seats is allocated to each party through allocateSeats().

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Find_Winners_CPL_16 **Name(s) of Testers: Shunichi Sawamura**

Test Description:
"findWinners" method:
Tests whether the find winners method in Election class will correctly save all candidates who obtained a seat in the election into the winnerList.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestCPL.java

Automated: yes_x___ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:

There must be at least 1 ballot in the given file.
Since findWinners() method is always implemented after completing voteCounting() method, the test runs with the condition that voteCounting() is already completed.
The winner list is an empty array list before running findWinners().
"testCPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compare the size of the winner list with the total seats given in the election	TestCPL.java testCPLVote.csv	The size of the winner list: 3	The size of the winner list: 3	The result matches the expected result.
2	Check if the winner list correctly has candidate elements as expected.	TestCPL.java testCPLVote.csv	The winner list stores the following candidate information: - (Name, Party, NumVote) - Pike, Democratic, 2 - Etta, Republican, 2 - Alawa, Republican, 2	The winner list stores the following candidate information: - (Name, Party, NumVote) - Pike, Democratic, 2 - Etta, Republican, 2 - Alawa, Republican, 2	The result matches the expected result.

Post condition(s) for Test:

The findWinners() figures out who is the winner in each party and saves all winners in a list.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit _x_ System __

Test Date: 3/24/2024

Test Case ID#: Find_Winners_OPL_17

Name(s) of Testers: Shunichi Sawamura

Test Description:
"findWinners" method:
Tests whether the find winners method in OPL class will correctly save all candidates who obtained a seat in the election into the winnerList.

Automated: yes_x__ no ___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestOPL.java

Results: Pass __x___ Fail_____

Preconditions for Test:

There must be at least 1 ballot in the given file.
Since findWinners() method is always implemented after completing voteCounting() and allocateSeats() methods, the test runs with the condition that voteCounting() is already completed.
The winner list is an empty array list before running findWinners().
"testOPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if the winner list has the winner from Democratic.	TestOPL.java testOPLVote.csv	Winner list has the following candidate information. - (Name, Party, NumVote) - Pike, Democratic, 2	Winner list has the following candidate information. - (Name, Party, NumVote) - Pike, Democratic, 2	The winner has the largest vote in the party. The result matches the expected result.
2	Check if the winner list has the winner from Republican.	TestOPL.java testOPLVote.csv	Winner list has the following candidate information. - (Name, Party, NumVote) - Alawa, Republican, 2	Winner list has the following candidate information. - (Name, Party, NumVote) - Alawa, Republican, 2	The winner has the largest vote in the party. The result matches the expected result.

Post condition(s) for Test:

The findWinners() figures out who is the winner in each party and saves all winners in a list.

Project Name: Project 1: Voting System — Team#16

Test Stage: Unit _x_ System __ **Test Date: 3/24/2024**

Test Case ID#: Find_Winners_CPL_Coin_Toss_18 **Name(s) of Testers: Shunichi Sawamura**

Test Description:
“findWinners” method:
Tests whether the find winners method in OPL class will correctly save all candidates who obtained a seat in the

election and run the coin toss if there is a tie between candidates.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Project1/src/TestOPL.java

Automated: yes x no

Results: Pass __x___ Fail_____

Preconditions for Test:

There must be at least 1 ballot in the given file.
The winner list is an empty array list before running findWinners().
"testOPLVote.csv" file should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if the winner list has the candidate from Republican.	TestOPL.java testOPLVote.csv	Winner list has one of the following candidate information. - (Name, Party, NumVote) - Etta, Republican, 2 - Alawa, Republican, 2	Winner list has one of the following candidate information. - (Name, Party, NumVote) - Etta, Republican, 2 - Alawa, Republican, 2	The winner is either Etta or Alawa because the winner is determined by coin toss. The result matches the expected result.
2	Check if the size of the winner list is equal to the total number of seat in the election.	TestOPL.java testOPLVote.csv	The size of the winner list: 1	The size of the winner list: 1	The result matches the expected result.

Post condition(s) for Test:

The findWinners() figures out who is the winner in each party and saves all winners in a list.

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit __ System _x_

Test Date: 3/24/2024

Test Case ID#: System_Testing_19

Name(s) of Testers: Shunichi Sawamura

Test Description:
Test the whole process correctly works. This test also covers testing Main object in "Main.java" because the whole process is controlled by Main.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manually Testing

Automated: yes___ no _x_

Results: Pass Fail x

Preconditions for Test:

There must be at least 1 ballot in the given file.
There must be at least 1 party in the given file.
There must be at least 1 candidate in the given file.
There must be more ballots than seats in the given file.
The input ballot file is correctly formatted and has no errors.
All testing files should move to the directory where the tester runs.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if the program asks the file name, and the user can put the filename.	Main.java testCPL.csv	Printed statement: "Please enter your file name:" User can type something after the output.	Printed statement: "Please enter your file name: " User can type something after the output.	The result matches the expected result.
2	Check if the user can provide the ballot file name from the command line argument	Main.java testCPL.csv	Program receives the file from the command line argument.	Program receives the file from the command line argument.	Example command line: java src/Main testCPL.csv The result matches the expected result.
3	Check if the program output the message when the input file is not found.	Main.java testCPL.csv	Printed statement: "File Not Found"	Printed statement: "File Not Found"	The result matches the expected result.
4	Check if the program repeatedly ask the file name if the file is not found.	Main.java testCPL.csv	Printed statement: "Please enter your file name:" User can type something after the output.	Printed statement: "Please enter your file name:" User can type something after the output.	The result matches the expected result.
5	Check if the program output the message when the input file is not a ballot file	Main.java Buglist.pdf	Printed statement: "Inappropriate File Format"	Error: Exception in thread "main" java.lang.NumberFormatException	FAIL: Error in this test.
6	Check if the program output the message when the input is a directory name	Main.java testing	Printed statement: "Inappropriate File Format"	Printed statement: "File Not Found" and Ended the process.	FAIL: The result is different from what we expected.
7	Check if the program displays	Main.java	----Election Results----	----Election Results----	FAIL: It prints out all the

	the election results after receiving the correct CPL ballot file	DisaplayResults.java testCPL.csv	Election type: CPL Number of Parties: 6 Number of Candidates: 11 Number of Seats: 3 Number of Ballots: 9 Number of Quota : 3 Democratic: Number of Seats: 1 *** Winner(s): Joe *** Number of Votes: 3 % of votes: 33.33333333333333 Candidate(s): Joe, Sally, Ahmed, _____ Republican: Number of Seats: 1 *** Winner(s): Allen *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Allen, Nikki, Taihui, _____ New Wave: Number of Seats: 0 *** Winner(s): N/A *** Number of Votes: 0 % of votes: 0.0 Candidate(s): Sarah, _____ Reform: Number of Seats: 1 *** Winner(s): Xinyue *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Xinyue, Nikita, _____ Green: Number of Seats: 0 *** Winner(s): N/A *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Bethany, _____ Independent: Number of Seats: 0 *** Winner(s): N/A *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Mike, _____	Election type: CPL Number of Parties: 6 Number of Candidates: 11 Number of Seats: 3 Number of Ballots: 9 Number of Quota : 3 Democratic: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 3 % of votes: 33.33333333333333 Candidate(s): Joe, Sally, Ahmed, _____ Republican: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Allen, Nikki, Taihui, _____ New Wave: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 0 % of votes: 0.0 Candidate(s): Sarah, _____ Reform: Number of Seats: 1 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 2 % of votes: 22.22222222222222 Candidate(s): Xinyue, Nikita, _____ Green: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Bethany, _____ Independent: Number of Seats: 0 *** Winner(s): Joe, Allen, Xinyue *** Number of Votes: 1 % of votes: 11.11111111111111 Candidate(s): Mike, _____	winners instead of the name of the candidate within that party.

			End the Process	End the Process	
8	Check if the audit file is generated after the program runs.	Main.java Audit.java testCPL.csv auditFile{Date_Time}.txt	auditFile{Date_Time}.txt is generated.	auditFile{Date_Time}.txt is generated.	The result matches the expected result. The file name is “auditFile” + date and time when the program generates an audit file.
9	Check if the generated audit file is readable but not writable	Main.java Audit.java testCPL.csv auditFile{Date_Time}.txt	Readable: True Writable: False	Readable: True Writable: False	The result matches the expected result.
10	Check if the file type of the audit file is txt.	Main.java Audit.java testCPL.csv auditFile{Date_Time}.txt	File type: txt	File type: txt	The result matches the expected result.
11	Check if the audit file saves the election information including results for CPL.	Main.java Audit.java testCPL.csv auditFile{Date_Time}.txt	Election type: CPL Number of Parties: 6 Number of Ballots: 9 Number of Seats: 3 Quota Value: 3 Democratic: Joe, Sally, Ahmed, Republican: Allen, Nikki, Taihui, New Wave: Sarah, Reform: Xinyue, Nikita, Green: Bethany, Independent: Mike, ----- Democratic ----- Total Seats: 1 Votes:3 / Quota:3 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 % of Vote to % of Seats: 33% / 33% ----- Republican ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 % of Vote to % of Seats: 22% / 33% ----- New Wave ----- Total Seats: 0 Votes:0 / Quota:3 = First Allocation Seats:0 Remaining Votes:0 --> Second Allocation Seats:0 % of Vote to % of Seats: 0% / 0% ----- Reform ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0	Election type: CPL Number of Parties: 6 Number of Ballots: 9 Number of Seats: 3 Quota Value: 3 Democratic: Joe, Sally, Ahmed, Republican: Allen, Nikki, Taihui, New Wave: Sarah, Reform: Xinyue, Nikita, Green: Bethany, Independent: Mike, ----- Democratic ----- Total Seats: 1 Votes:3 / Quota:3 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 ----- Republican ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- New Wave ----- Total Seats: 0 Votes:0 / Quota:3 = First Allocation Seats:0 Remaining Votes:0 --> Second Allocation Seats:0 ----- Reform ----- Total Seats: 1 Votes:2 / Quota:3 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:1 ----- Green -----	FAIL: % of Vote to % of Seats is not saved in the generated audit file.

			Remaining Votes:2 --> Second Allocation Seats:1 % of Vote to % of Seats: 22% / 33% ----- Green ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 % of Vote to % of Seats: 11% / 0% ----- Independent ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 % of Vote to % of Seats: 11% / 0% *** Winner(s) *** Joe (Democratic) Allen (Republican) Xinyue (Reform)	Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 ----- Independent ----- Total Seats: 0 Votes:1 / Quota:3 = First Allocation Seats:0 Remaining Votes:1 --> Second Allocation Seats:0 *** Winner(s) *** Joe (Democratic) Allen (Republican) Xinyue (Reform)	
12	Check if the program can handle the huge ballot CPL file that has 100,000 ballots in 4 minutes.	Main.java testCPLLong.csv	Running Time: less than 4 minutes	Running Time: less than 4 minutes	The result matches the expected result.
13	Check if the program can correctly distribute the seats to parties in CPL if the number of seats is more than the number of parties.	Main.java testCPLSeatsMoreThanParty2.csv	Winner(s): Joe, Sally, Ahmed, Dale, Allen, Nikki, Taihui	Winner(s): Joe, Sally, Ahmed, Dale, Allen, Nikki, Taihui, Nick	FAIL: Too many candidates win. (Number of seats: 7, Number of winners: 8). Also, the number of total allocated seats become 10 while it should be 7.
14	Check if the program displays the election results after receiving the correct OPL ballot file	Main.java DisaplayResults.java testOPL.csv	----Election Results---- Election type: OPL Number of Parties: 3 Number of Candidates: 6 Number of Seats: 2 Number of Ballots: 9 Number of Quota : 4 ***** Winner ***** 1. Pike (% of number of toal votes 22.22222222222222 | number of votes 2) 2. Etta (% of number of toal votes 22.22222222222222 | number of votes 2) ***** Candidate ***** Democrat Won: 1 seat(s) Candidate: Pike, Lucy, Beiye _____ Republican Won: 1 seat(s) Candidate: Etta, Alawa _____ Independent1 Won: 0 seat(s) Candidate: Sasha	----Election Results---- Election type: OPL Number of Parties: 3 Number of Candidates: 6 Number of Seats: 2 Number of Ballots: 9 Number of Quota : 4 ***** Winner ***** 1. Pike (% of number of toal votes 22.22222222222222 | number of votes 2) 2. Etta (% of number of toal votes 22.22222222222222 | number of votes 2) ***** Candidate ***** Democrat Won: 1 seat(s) Candidate: Pike, Lucy, Beiye _____ Republican Won: 1 seat(s) Candidate: Etta, Alawa _____ Independent1 Won: 0 seat(s) Candidate: Sasha	The result matches the expected result.

			End the Process	End the Process	
15	Check if the audit file saves the election information including results for OPL.	Main.java Audit.java testOPL.csv	Election type: OPL Number of Parties: 3 Number of Ballots: 9 Number of Seats: 2 Quota Value: 4 Democrat: Pike, Lucy, Beiye, Republican: Etta, Alawa, Independent1: Sasha, ----- Democrat ----- Total Seats: 1 Votes:3 / Quota:4 = First Allocation Seats:0 Remaining Votes:3 --> Second Allocation Seats:1 Pike Votes: 2 Lucy Votes: 1 Beiye Votes: 0 ----- Republican ----- Total Seats: 1 Votes:4 / Quota:4 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 Etta Votes: 2 Alawa Votes: 2 ----- Independent1 ----- Total Seats: 0 Votes:2 / Quota:4 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:0 Sasha Votes: 2 *** Winner(s) *** Pike (Democrat) Etta (Republican)	Election type: OPL Number of Parties: 3 Number of Ballots: 9 Number of Seats: 2 Quota Value: 4 Democrat: Pike, Lucy, Beiye, Republican: Etta, Alawa, Independent1: Sasha, ----- Democrat ----- Total Seats: 1 Votes:3 / Quota:4 = First Allocation Seats:0 Remaining Votes:3 --> Second Allocation Seats:1 Pike Votes: 2 Lucy Votes: 1 Beiye Votes: 0 ----- Republican ----- Total Seats: 1 Votes:4 / Quota:4 = First Allocation Seats:1 Remaining Votes:0 --> Second Allocation Seats:0 Etta Votes: 2 Alawa Votes: 2 ----- Independent1 ----- Total Seats: 0 Votes:2 / Quota:4 = First Allocation Seats:0 Remaining Votes:2 --> Second Allocation Seats:0 Sasha Votes: 2 *** Winner(s) *** Pike (Democrat) Etta (Republican)	The result matches the expected result.
16	Check if the program can handle the huge ballot OPL file that has 100,000 ballots in 4 minutes.	Main.java testOPLLong.csv	Running Time: less than 4 minutes	Running Time: less than 4 minutes	The result matches the expected result.

Post condition(s) for Test:

The whole process is completed, and the audit file is saved in the directory where the tester runs the program.